Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method in a computer system for transferring data, comprising:

receiving the data in parallel and demultiplexing the data into separate data units;

~~providing~~ using a socket to create a virtual connection to transfer data between a first application process residing on a server computer and a process residing on a client computer of a computer network; ~~and~~

assigning and binding a thread to an available socket;

releasing the socket ~~when~~ as soon as the data has finished transferring to allow the virtual connection to transfer data between a second application process and the process residing on the client computer of the computer network; and

reusing at least one socket and allowing the thread to serve at least one socket and transmit data through the thread's corresponding socket in parallel.

2. (Currently Amended) The method as set forth in claim 1, further comprising ~~providing~~ using a plurality of sockets to create additional virtual connections between application processes and processes on plural client computers of the computer network.

3. (Original) The method as set forth in claim 2, further comprising assigning each of the application processes to an available one of the plurality of sockets.

4. (Original) The method as set forth in claim 3, wherein each of the plurality of sockets is determined not to be in use prior to the assigning.

5. (Original) The method as set forth in claim 3, wherein assigning is performed using at least one of the following assignment techniques: (a) round robin; (b) random; (c) user defined.

6. (Original) A computer-readable medium having computer-executable instructions for performing the method as set forth in claim 1.

7. (Currently Amended) A method in a computer system for transferring data between a computer having a processor and a client computer of a computer network, comprising:

receiving the data in parallel and demultiplexing the data into separate data units;

creating a plurality of sockets capable of ~~providing~~ using virtual connections between processes executing on the processor and a process residing on the client computer of the computer network; ~~and~~

assigning each of the processes to an available socket of the plurality of sockets in response to a request for a socket;

releasing at least one socket as soon as associated data has been transferred; and

reusing at least one socket and allowing the processes to serve at least one socket and transmit data in parallel through the socket of the corresponding process.

8. (Original) The method as set forth in claim 7, wherein assigning is performed using a round robin socket assignment technique.

9. (Original) The method as set forth in claim 7, wherein assigning is performed using a random socket assignment technique.

10. (Original) The method as set forth in claim 7, wherein assigning is performed using a user-defined assignment technique.

11. (Original) The method as set forth in claim 7, wherein the data is divided into separate data units.

12. (Original) The method as set forth in claim 7, wherein the data are incoming network requests and the incoming network requests are demultiplexed into separate network requests corresponding to a data unit.

13. (Currently Amended) A data transfer system in a computer system for transferring network data, comprising:

a demultiplexer that receives the data in parallel and demultiplexes it into separate data units;

a plurality of sockets for ~~providing~~ using a virtual connection between a process residing on a server computer and a process residing on a client computer of a computer network;

a plurality of threads for processing the network data, each one of the plurality of threads capable of being assigned one of the plurality of sockets; and

a release module that releases at least one socket as soon as associated data has been transferred and allows reuse of the released socket for enabling an associated thread to serve at least one socket and transmit data in parallel through the socket of the corresponding thread; and

a parallel sockets module in communication with the plurality of sockets and the plurality of threads that provides parallel transfer of the network data using the plurality of sockets.

14. (Original) The data transfer system as set forth in claim 13, wherein the parallel sockets module further comprises a network data processor that divides the network data into a plurality of data units.

15. (Original) The data transfer system as set forth in claim 13, wherein the parallel sockets module further comprises an assignment module that uses a socket assignment technique to assign at least one of plurality of threads to an available one of the plurality of sockets.

16. (Original) The data transfer system as set forth in claim 15, wherein the parallel sockets module further comprises a binding module that binds the assigned thread to the assigned socket.

17. (Original) The data transfer system as set forth in claim 15, wherein the socket assignment technique is a round robin technique that assigns the thread to a first available socket.

18. (Original) The data transfer system as set forth in claim 15, wherein the socket assignment technique is a random technique that assigns the thread randomly to an available one of the plurality of sockets.

19. (Original) The data transfer system as set forth in claim 15, wherein the socket assignment technique is a user-defined technique that assigns the thread to an available one of the plurality of sockets as determined by a user.

20. (Currently Amended) A computer-implemented method for transferring data between a network server and a client computer of a computer network, comprising:

receiving the data in parallel and demultiplexing the data into separate data units;

~~providing~~ using a socket having a virtual connection between a first server process on the network server and a process residing on the client computer of the computer network;

determining that the socket is available;

binding the first server process to the available socket to facilitate the transfer of data; ~~and~~

releasing at least one socket as soon as associated data has been transferred; and

making the socket available to a second server process for reuse when the data has finished transferring and allowing the processes to serve at least one socket and transmit data in parallel through the socket of the corresponding process.